PolyOrBAC: a security framework for critical infrastructures

Anas Abou El Kalam, Yves Deswarte, Amine Baïna, Mohamed Kaâniche

PolyOrBAC: A Security Framework for Critical Infrastructures

A. Abou El Kalam[1], Y. Deswarte[2,3], A. Baïna[2,3], M. Kaâniche[2,3]

(1) Université de Toulouse ; IRIT ; INPT-ENSEEIHT ; 2 rue Camichel, F-31071 Toulouse, France
(2) CNRS ; LAAS ; 7 avenue du Colonel Roche, F-31077 Toulouse, France
(3) Université de Toulouse ; UPS, INSA, INP, ISAE ; LAAS ; F-31077 Toulouse, France

Abstract

Due to physical and logical vulnerabilities, a critical infrastructure (CI) can encounter failures of various degrees of severity, and since there are many interdependencies between CIs, simple failures can have dramatic consequences on the users. In this paper, we mainly focus on malicious threats that might affect the information and communciation system that controls the Critical Infrastructure, i.e., the Critical Information Infrastructure (CII). To address the security challenges that are specific of CIIs, we propose a collaborative access control framework called PolyOrBAC. This approach offers each organization taking part in the CII the capacity of collaborating with the other ones, while maintaining a control on its resources and on its internal security policy. The interactions between organizations participating in the CII are implemented through web services (WS), and for each WS a contract is signed between the service-provider organization and the service-user organization. The contract describes the WS functions and parameters, the liability of each party and the security rules controlling the interactions. At runtime, the compliance of all interactions with these security rules is checked. Every deviation from the signed contracts triggers an alarm, the concerned parties are notified and audits can be used as evidence for sanctioning the party responsible for the deviation. Our approach is illustrated by a practical scenario, based on real emergency actions in an electric power grid infrastructure, and a simulation test bed has been implemented to animate this scenario and experiment its security issues.

Key words: Critical Infrastructure, Security, Access Control Policies and Models, Collaboration, Interoperability.

1. Introduction

Critical Infrastructures (CI) are physical and logical facilities of essential importance for public welfare. Their failure or disruption could potentially have a dramatic impact on economic and social welfare of a nation, a society, or an economy. Significant examples of CIs include those dedicated to electricity generation, transport and distribution (i.e., the electric power grid), telecommunications, supply services (energy, food, fuel, water, gas), transportation systems (roads, railways, airports), financial services (banks, stock exchange, insurances), etc.

Due to interdependencies between various infrastructures, cascading failures[1] and escalating failures[2] are not unlikely [1], [2]. A simple failure can propagate at a large scale as in the example of the North America blackout occurred on 14 august 2003 [3]. One of the immediate causes of this large blackout, which caused 6 billion dollars of losses according to the US Department of Energy, was a failure of the monitoring software, which prevented confining an electrical line incident before its propagation across the electrical power grid. Such failure scenarios might occur as a result of accidental faults as well as of malicious threats (intrusions, worm propagation, denial of service attacks, etc.).

A CI is controlled by an Information and Communication system, called Critical Information Infrastructure (CII). This CII uses potentially vulnerable information technologies, which can be targetted and compromised by malicious attackers. This is why ensuring the security of CIIs has become an important topic of many research studies.

To deal with the changing needs of the market, the CII must be flexible and extensible (co-operation with new organizations, possibly on new geographical areas). Consequently, the CII must be open and distributed to allow the organizations participating in it to collaborate and provide globally resources and services to users belonging to the various organizations. With the opening and the deregulation of markets, some of these organizations can

[1] A cascading failure occurs when a failure in one infrastructure causes the failure of one or more components in a second infrastructure.

[2] An escalating failure is a failure in one infrastructure that is exacerbated by an independent failure in another infrastructure, generally in the form of increasing the severity or the time for recovery or restoration of the first failure.

be in competition, while collaborating. It is the case in Europe in particular, where regional, national or multinational companies are in competition but must cooperate to produce, transport and distribute electric power. This induces an essential need for access control policies and enforcing mechanisms, to manage different accesses from an organization to the other.

The work presented here only focuses on security challenges related to access control, collaboration, and interoperability between the organizations and the systems composing the CII [4], [5], [6]. The novelty of this paper is to present a global and homogeneous approach, called PolyOrBAC, to ensure security in the CII. In particular, we (1) define the local policies of the organizations participating to the CII, (2) extend OrBAC with notions such as virtual users and service images to control interactions between organizations through e-contracts, (3) show how e-contracts can be expressed by security rules and checked at run time to detect dynamic violations and abuses, and finally (4) present our implementation and illustrate our global approach through concrete examples.

This paper is thus gathering in a consistent way several works published previously [4], [5], [6] [7], and extends them with more information on how the PolyOrBAC framework can be applied to CIIs and on how it can be implemented. Especially sections 4, 5 and annexes are new.

The remainder of this paper is organized as follows: Section 2 identifies the global security requirements of CIIs and confront these requirements to traditional access control models. Then a sketch of our proposal is presented in Section 3. In particular, we progressively demonstrate how PolyOrBAC integrates WS, OrBAC and e-contracts to meet the CII's security requirements. Section 4 shows how PolyOrBAC can be applied to the electrical power grid CII. Section 5 presents details of our test bed and Section 5 draws up the conclusions and proposes possible extensions of this work.

2. CII Security requirements and related works

2.1. CII Security requirements

Globally, a CI can be seen as a set of interconnected organizations involving different actors and stakeholders (e.g., power generation companies, substations, energy authorities, maintenance service providers, transmission and distribution system operators, etc.) using heterogeneneous logical and physical information and communication systems and networks, exhibiting different levels of security threats and protection mechanisms.

In order to provide a satisfactory level of protection for the global interconnected critical infrastructure, the following security constraints and requirements need to be carefully addressed.

1. *Secure cooperation* between different organizations, possibly mutually suspicious, with different features, functioning rules and policies. In this context, we should secure not only intra-organizational accesses but also interactions between different independent organizations.
2. *Audit and assessment*: especially for inter-organizational workflows, the audit should determine if the protections which are defined in the policy are being correctly enforced in practice; it also keeps logs on interactions between partners, to verify if they comply with the previously signed contracts, and provide evidence in case of dispute or dynamic abuses.
3. *Autonomy and loose coupling*: each organization controls its own security policy, users, resources, applications, etc., while respecting the global functioning and security rules of the whole system.
4. *Enforcement of permission, explicit prohibition as well as obligation rules*: explicit prohibitions can be particularly useful, as we have decentralized policies where each administrator does not know details about the other parts of the infrastructure. Moreover, explicit prohibitions can also specify exceptions, or limit the propagation of permissions in case of role hierarchies. Similarly, obligations can be useful to impose some actions that should be carried out by users or that should be automatically performed by the system itself.

2.2. Related works

To satisfy the CII security requirements cited above, two global approaches are possible, centralized or Peer-to-Peer. In the following, we discuss some examples of significant contributions in each of these categories.

2.2.1. Centralized approaches

Several works on Workfow Management Systems follow the centralized approach. For example, Bertino *et al.* describe different configurations and constraints associated to the workflow execution [8]. Alturi *et al.* use colored and timed petri nets to define a conceptual and a logical workflow authorization model and to enforce the authorization flow based on the interdependencies between activities [9]. However, these works do not show how

to enforce inter-organization workflows while respecting the global and local constraints, especially when organizations collaborate to achieve a common objective. Moreover, these works suppose the existence of a central entity responsible for specifying and managing the workflow security policy without showing how such a policy will be used. Hence, applied to our context, these solutions imply the definition of a global security policy for the CII, to which all participating organizations must adapt their own security policies to conform with this global policy. Finally, these solutions do not show how the security policy can be dynamically monitored during the workflow execution while this requirement is important in our context. In fact, we believe that this centralized approach is generally unacceptable for large companies that may involve several independent CIIs, which could potentially define conflicting security policies. Lin, Rao and Bertino [10], proposed a novel policy-based access control model for collaborative access control, the main idea is based on the notion of policy decomposition. This architecture is developed based on the XACML framework [29] which allows the proposed solution to be easily integrated into existing systems. It also presents algorithms for decomposing a global policy that is enforced by a set of collaborating parties without compromising the autonomy or confidentiality requirements of the collaborating parties and to efficiently evaluate requests from different parties. While very interesting, this work cannot be directly applied in our context as it imposes a global access control policy over a collaborative environment; this constraint involves information-disclosure while in CII we have mutually suspicous organizations that should keep secret some details of their policies. Jajodia, and Samarati [11] presented a unified framework that can enforce multiple access control policies within a single system. The framework is based on a language through which users can specify security policies to be enforced on specific accesses. The language allows the specification of both positive and negative authorizations and incorporates notions of authorization derivation, conflict resolution, and decision strategies. Different strategies may be applied to different users, groups, objects, or roles, based on the needs of the security policy. The major advantage of this approach is that it can be used to specify different access control policies that can be enforced by the same security server. However it does not take into consideration collaboration and autonomy issues, which are important in the context of CIIs. In term of languages, significant contributions are related to Ponder [32] and XACML. For example, Lorch, Proctor *et al.* present XACML, a standard access control language, as one component of a distributed and

inter-operable authorization framework [12]. This work illustrates how authorization can be deployed in distributed, decentralized systems, and helps connecting the general components of an authorization system. XACML is useful for specifying complex policies in a wide variety of distributed applications, environments and systems. However, it has some limitations, the language flexibility and expressiveness comes at the cost of complexity and verbosity. In practice, using this laguage is painful as it leads to complex policy files. Tools are underway, but as long as they are not wideky available, it will be hard for average users to work with any XACML-based system. And even with good tools in place, there is an inherent semantic complexity that accumulates over the syntactic complications.

2.2.2. Peer-to-Peer approaches

Peer-to-peer solutions have been proposed in some previous works. These approaches do not assume the existence of a global CII organization. In 2008, Sturm and Dittrich [13] presented a fine grained access control mechanism for peer-to-peer collaborations. This mechanism is based on the local access control components of the participants. The peers export their access control policies in XACML. Two mechanisms are proposed to combine these policies. The first approach establishes mappings between the export policies. The second approach installs a distributed access control directory. While mappings are created between two peers, a directory contains all rights of all users of all peers. In a similar work, Shehab and Bertino [14] presented a distributed secure interoperability framework for mediator-free collaboration environments in which domains collaborate in making localized access control decisions. This work introduced the idea of secure access paths which enables domains to make localized access control decisions without having a global view of the collaboration. It also presented a path authentication technique for proving path authenticity. Basically, it presented proactive and on-demand path discovery algorithms that enable domains to securely discover paths in the collaboration environment. Pearlman and Welch [15] presented the Community Authorization Service (CAS) intended to solve three critical authorization problems that arise in distributed virtual organizations: scalability, flexibility and expressibility, and the need for policy hierarchies. Their work addressed these problems by introducing a trusted third party administrated by the virtual organization that performs fine-grain control of community policy while leaving ultimate control of resource access with the resource owners. This work is interesting in the fact that it

presents an authorization framework for distributed and collaborative environments. However, it uses a third party with a global knowledge of policies of other organizations. This is a limitation that our work tries to overcome. In the same way, MultiOrBAC [16] and O2O [17] stipulate that the various organizations accept to cooperate so that roles in one organization are given privileges in another organization. For that, each participating organization must trust the others, at least for the definition of their roles and for the assignment of the corresponding roles to trustworthy users. This approach is also intrusive with respect to the confidentiality of each organization's internal structure, user identity, and security policy. This is equally unacceptable for CIIs where participating organizations are in competition, and thus are mutually suspicious. Clearly, each organization wants to keep its autonomy on the choice of its internal security policy, and would not accept to open its information and communication system to unknown external users working for its competitors. Ideally, an organization should know nothing about the other organizations' users or assets, but only the information needed to cooperate fairly. Enabling a secure collaboration between organizations while preserving each organization's autonomy and self-determination is the challenge addressed by our approach, the PolyOrBAC framework, presented in the next section.

3. The PolyOrBAC security framework

Based on an access and flow control model extended from OrBAC [18] and on web services mechanisms, PolyOrBAC defines, deploys and audits a security policy both in intra and inter-organizational workflows. It mainly gives answers to questions such as: how to define intra-organizational accesses? how to specify inter-organizational access policies? how to specify and deploy e-contracts that can be agreed between organizations collaborating within a CII? how can we enforce access control and be able to audit and detect possible runtime violations and abuses? The following subsections describe our proposal. First, we show how specifying local security policies with OrBAC (inside each organization). Then, we introduce new notions (virtual users and WSs images) to manage inter-organizational accesses. Finally,, we use e-contract to express and runtime check WSs interactions.

3.1. Specifying local security policies with OrBAC

In the context of CIIs, each organization specifies its own security policy, which defines who has access to what, when, and in which conditions. In this subsection, we show that OrBAC is the most suitable access control model for achieving this task. Note that for this paper to be self-contained and to situate this work regarding the state of the art, we recall the main notions of OrBAC and some other related access control models.

Actually, the OrBAC (*Organization-based Access Control*) model is an extension of the traditional RBAC (*Role-Based Access Control*) model [19], [20]. In RBAC, roles are assigned to users, permissions are assigned to roles and users acquire permissions by playing roles. By abstracting users into roles, RBAC facilitates the security policy management. Indeed, if users are added to or are withdrawn from the system, only instances of the relationship between users and roles need to be updated.

OrBAC goes further by abstracting objects into views and actions into activities. In this way, security rules are specified by abstract entities only. Consequently, the representation of the security policy is completely separated from the implementation.

More precisely, in OrBAC, an activity is a group of one or more actions; a view is a group of one or more objects; and a context is a specific situation that conditions the validity of a rule. Actually, two security levels can be distinguished in OrBAC:

- *`Abstract level`*: the administrator defines security rules through abstract entities (roles, activities, views) without worrying about how each organization implements these entities.

- *`Concrete level`*: when a user requests to perform an action on an object, permissions are granted to him according to the concerned rules, the organization, the role currently played by the user, the requested action (that instantiates an activity defined in the rule) on the object (that instantiates a view defined in the rule), and the current context. The derivation of permissions (i.e., runtime evaluation of security rules) can be formally expressed as follows:

$\forall$ org $\in$ Organizations, $\forall$s $\in$ Subjects, $\forall$ $\alpha \in$ Actions, $\forall$ o $\in$ Objects,
$\forall$ r $\in$ Roles, $\forall$a $\in$ Activities, $\forall$ v $\in$ Views, $\forall$ c $\in$ Contexts
***Permission* (org, r, v, a, c)** $\wedge$
Empower (org, s, r) $\wedge$
Consider (org, α, a) $\wedge$
Use (org, o, v) $\wedge$
Hold (org, s, a, o, c)
$\rightarrow$ ***Is permitted*(s, α, o)**

This rule means: if in a certain organization, a security rule specifies that role r can carry out the activity a on the view v when the context c is *true*, and if r is assigned to subject s, if action α is a part of a, and if object o is part of v, and if c is *true*, then s is allowed to perform α (e.g., WRITE) on o (e.g., F1.TXT). Prohibitions, and obligations can be defined in the same way.

As rules are expressed only through abstract entities, OrBAC is able to specify the security policies of several collaborating and heterogeneous sub-organizations (e.g., departments) of a "global organization". In fact, the same role, e.g., OPERATOR can be played by several users belonging to different sub-organizations; the same view e.g., "TECHNICALFILE", can designate a table TF-TABLE in one sub-organization or a XML object TF1.XML in another one; and the same activity READ can correspond in a particular sub-organization to a SELECT action while in another sub-organization it may specify an OPENXMLFILE() action.

In our context, OrBAC presents several benefits and satisfies several security requirements of organizations participating in a CII: rules expressiveness, abstraction of the security policy, scalability, heterogeneity and evolvability. OrBAC is thus more suitable than RBAC (and its variants); in particular, for specifying local security policies of the CII's organizations. These security policies can subsequently be locally enforced by (local) security mechanisms such as Access Control Lists (ACL), firewall rules, security credentials (e.g., XML capabilities), OASIS WS security mechanisms, etc.

3.2. Managing interactions between organizations

While OrBAC is suitable for specifying local security policies, it has an important limitation in our context. Actually, OrBAC is centralized and does not handle collaborations between non-hierarchical organizations, i.e.,

secure handling of accesses to external resources. In fact, OrBAC is not appropriate for specifying rules that involve several autonomous organizations. Moreover, it is not possible to associate permissions to users belonging to other organizations. As a result, OrBAC is unfortunately only adapted to centralized infrastructures and does not cover the distribution and collaboration needs of current CIIs. To fulfill this requirement, we have proposed the MultiOrBAC model in [16]. Basically, MultiOrBAC abstract rules specify that roles in a certain organization are permitted (or prohibited or obliged) to carry out activities on views belonging to other organizations. Therefore, contrarily to OrBAC, a MultiOrBAC rule may involve two different organizations that do not belong to the same hierarchy: the organization where the role is played, and the organization which the view and the activity belong to.

However, in the context of CIIs, MultiOrBAC presents several weaknesses. In fact, MultiOrBAC offers the possibility to define local rules / accesses for external roles (i.e., belonging to another organization), without having any information about who plays these roles and how the (*user*, *role*) association is managed in the remote organization. This causes a serious problem of responsibility and liability: who is responsible in case of remote abuse of privileges? How can the organization to which belongs the object have total confidence in the organization to which belongs the user?

The MultiOrBAC logic is thus not adapted to CIIs where in-competition organizations can naturally be mutually suspicious. Moreover, in MultiOrBAC the access control decision and enforcement are done by each organization, which means that the global security policy is in fact defined by the set of the organizations' security policies. In that case, it is difficult to enforce and maintain the consistency of the global security policy, in particular if each organization's security policy evolves independently.

In our PolyOrBAC framework, collaboration and interactions between organizations are made through the use of the WS technology, which provides platform-independent protocols and standards for exchanging heterogeneous interoperable data services. Software applications written in various programming languages and running on various platforms can use WS to exchange data over computer networks in a manner similar to inter-process communication on a single computer. WS also provide a common infrastructure and services for data access, integration, provisioning, cataloging and security. These functionalities are made possible through the use of open standards, such as: XML for exchanging heterogeneous data in a common

information format [21]; SOAP, a data transport mechanism to send data between applications in one or several operating systems [22]; WSDL, used to describe the services that a business offers and to provide a way for individuals and other businesses to access those services [23] and UDDI, an XML-based registry which enables businesses to list themselves and their services on the Internet and discover each other [24].

Note that some recent works already tried to combine web services mechanisms and security policies based on RBAC. Beznosov and Deng presented a framework for implementing Role-Based Access Control using CORBA security service [25]. Vuong, Smith and Deng proposed an XML-Based approach to specify enterprise RBAC policies [26]. In 2004, Feng, Guoyuan and Xuzhou suggested SRBAC, a Service-oriented Role-Based Access Control model and security architecture model for Web Services [27]; and Leune, Van and Heuvel presented RBAC4WS, a methodology for designing and developing a Role-Based Access Control model for Web Services [28]. Focusing on service invocation, this methodology adopts a symmetric perspective considering both the supplier and the customer. Besides, some other works tried to couple XACML with RBAC. For example, in 2004, OASIS adopted an XACML profile for Role Based Access Control, while in 2005, Crampton proposed an RBAC policy using an XACML formulation [29].

In the proposed PolyOrBAC framework, we integrate WS and OrBAC. To achieve this task, we introduce two new notions, the virtual users and the WS images:

- *for the organization offering a WS* (i.e., that allows external accesses to its local resources through WS interfaces), the other organization is seen as a virtual user which plays a role authorized to use the WS;
- *for the organization requesting the WS*, the WS is seen as an external object, locally represented by its image.

To illustrate these notions, we explain the two main phases of PolyOrBAC: (1) *publication and negotiation of collaboration rules as well as the corresponding access control rules* and (2) *runtime access to remote services.* In the first phase, each organization determines which resources it will offer to external partners. Web services are then developed on application servers, and referenced on the Web Interface to be accessible to external users.
When an organization publishes its WS at the UDDI registry, the other organizations can contact it to express their wish to use the WS. Let us take

a simple example where organization B offers WS1, and organization A is interested in using WS1. A and B should negotiate and come to an agreement concerning the use of WS1. Then, A and B establish a contract[3] and jointly define security rules concerning the access to WS1. These rules are registered (according to an OrBAC format) in databases located at both A and B. For instance, if the agreement between A and B is "users from A have the permission to consult B's measurements in the emergency context", B should, in its OrBAC security policy:

- have (or create) a rule that grants the permission to a certain (local) role (e.g., OPERATOR) to consult its measurements: *Permission*(B, *Operator*, *Measurements*, *Consult*, *Emergency*);
- create a VIRTUAL USER noted *PartnerA* that represents A for its use of WS1;
- add the *Empower*(*B*, *PartnerA*, *Operator*) association to its rule base. This rule grants *PartnerA* the right to play the OPERATOR role.

In parallel, A creates locally a *WS1_image* which (locally in A) represents WS1 (i.e., the WS proposed by B), and adds a rule in its OrBAC base to define which of A's roles can invoke *WS1_image* to use WS1.

Considering the second phase of PolyOrBAC dedicated to the control of runtime access to remote services, we use an AAA (*Authentication*, *Authorization* and *Accounting*) architecture, which separates authentication from authorization; we distinguish access control decision from access control enforcement; and we keep access logs in each organization. Basically, if a user from A (let us note it Alice) wants to carry out an activity, she is first authenticated by A. Then, protection mechanisms of A check if the OrBAC security policy (of A) allows this activity. We suppose that this activity contains local as well as external accesses (e.g., invocation of B's WS1). Local accesses should be controlled according to A's policy, while the WS1 invocation is both controlled by A's policy (Alice must play a role that is permitted to invoke *WS1_image*), and by B's policy (the invocation is transmitted to virtual user PartnerA, which must play a role authorized to execute the web

[3]The contract aspects will be discussed in the next subsection.

service), according to the contract established between A and B. If both policies grant the invocation, WS1 is executed (under the access control enforcement mechanisms implemented by A and by B).

3.3. Expressing and checking WS interactions with e-contracts

In the last subsection, we have shown that PolyOrBAC offers several useful concepts and mechanisms for access control in CIIs: it permits a better specification and control of local security policies through OrBAC; each organization authenticates its users and manages its resources autonomously; and interactions are handled by WS. Consequently, the service-requesting organization is liable for its users, and thus is responsible for the actions carried out by their users. Inversely, the service-providing organization is liable for its services. However, some aspects need to be addressed:

- Enforcement and real time checking of contracts established between different organizations; in fact, the system must be able to check the satisfaction as well as the correct enforcement of the signed contracts.

- Audit logging and assessment of the different actions: in fact, in large scale systems, experience has shown that even if the security policy is consistent (thanks to an off-line verification), violations and abuses (especially, remote abuse of priveleges) can occur dynamically at runtime. Hence, we need a mechanism that can detect this kind of dysfunctional behavior and to notify the concerned parties.

- Handling of mutual suspicion between organizations; no information is disclosed about local security policies; the organization providing the web service does not know which user of the other organization requests the web service, and the organisation requesting the web service does not know which role performs which activity in the service providing organization. It is also necessary to detect any abuse of the contract by a malicious organization.

To deal with these issues, we state that for each WS use, an e-contract should be negociated between the partner organizations (WS provider and WS client). This contract must specify precisely the web service functions and parameters (including the expected quality of service, the liability of each party, payment, penalties in case of abuse, etc.), and also the security rules related to the invocation and the execution of the web service. These

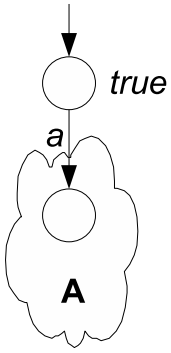

Figure 1: Modeling Permissions.

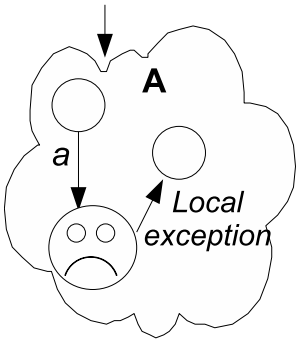

Figure 2: Modeling prohibitions.

security rules must be checked and enforced at runtime to prevent, or at least detect any abuse. The security rules can be expressed with a syntax close to OrBAC (see Section 3.3).

The question that arises now is how to specify e-contracts. Actually, the most relevant security requirements for contracts are workflows, actions, permissions, prohibitions, obligations, time constraints, disputes and sanctions.

To express these requirements, we propose using timed automata [30]. First, permissions (actions that are authorized by the contract clauses) are simply specified through transitions in the timed automata. For instance, in Fig. 1, the system can (i.e., has the permission to) execute the action a at any time and then, behaves like the automaton A.

Second, we distinguish two kinds of prohibitions in e-contracts:

- *Implicit prohibitions*: the idea is to only specify permissions in the automata; the states, actions and transitions not represented in the automata are by essence *prohibited* because the runtime model checker will not recognize them.

- *Explicit prohibitions*: explicit prohibitions can be particularily useful in the management of decentralized policies / contracts where each administrator does not have details about the other organizations participating in the CII. Moreover, explicit prohibitions can also specify exceptions or limit the propagation of permissions in case of hierarchies. In our model, we specify explicit prohibitions by adding a "failure state" where the system will be automatically led if a malicious action is detected. In Fig. 2, as the a action is forbidden, its execu-

tion automatically leads to the failure state described by an "unhappy face", which automatically triggers an exception carried out locally.

Let us now deal with obligations. Recently, several works have focussed on the modeling of this access modality [31] [32] [33] [34]. In XACML [35], obligations are a set of operations that must be fulfilled in conjunction with an authorization decision (permit or deny). Bettini *et al.* distinguish between provisions and obligations [31]. Provisions are conditions that need to be satisfied or actions that must be performed before a decision is rendered, while obligations are actions that must be fulfilled by either the users or the system after the decision. Hilty *et al.* define the OSL, an Obligation Specification Language, that allows formulating a wide range of usage control requirements [34]. They differentiate between usage and obligational formulae. Usage is concerned with operations (e.g., processing, rendering, execution, management, or distribution) on data that must be protected; while obligational formulae are conditions on the usage of data, e.g., "delete document D within 30 days". An obligational formula becomes an obligation once a data consumer is obliged to satisfy it, i.e., once the data consumer has received the data and committed to the condition.

In our vision, obligations are actions that "must" be carried out; otherwise the concerned entity will be subject to sanctions. Besides that, as every obligation is also a permission[4], obligations will be specified by particular transitions (in the same way as permissions). However, as obligations are stronger than permissions, we should add another symbols to capture this semantics and to distinguish between what is mandatory and what is permitted but not mandatory. Actually, to model obligations, we use transition time-outs and invariants.

In this respect, an obligation is considered as a simple transition, and if a maximum delay is assigned to the obligation, a *time-out* (noted by d in Fig. 3) is set for the delay. When the obligation is fulfilled, this event resets the time-out and the system behaves like A1. On the contrary, if the time-out expires, an exception is raised and the system behaves like A2 (which can be considered as an *exception*).

Basically, when an explicit prohibition occurs when an obligation is not fulfilled, a conflicting situation (e.g., one of the parties does not comply

[4]A mandatory action should be permitted; in other words, we cannot render mandatory something that is not permitted.

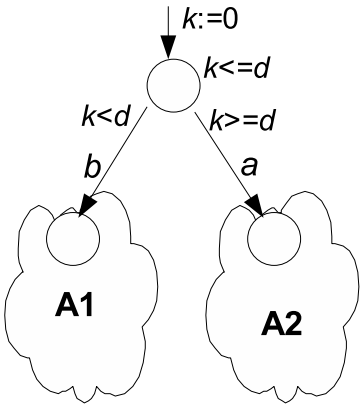

Figure 3: Modeling obligations.

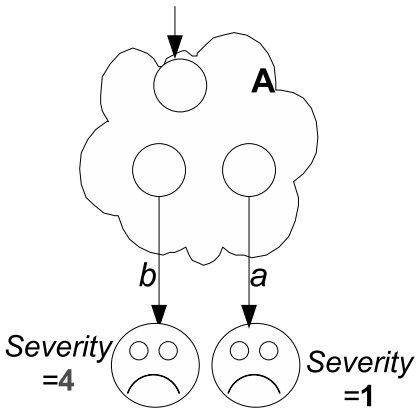

Figure 4: Modeling dispute situations.

with the contract clauses) arises, and the automaton automatically makes a transition to a dispute situation (i.e., to the unhappy state) or triggers an exception processing (A2 in Fig. 3). Actually, modeling disputes will allow to not only identify anomalies and violations, but go further by identifying activities (succession of actions, interactions) that led to these situations, and finally can automatically lead to the cancelation of the contract. Moreover, as disputes have different severities and as they are not all subject to the same sanctions, we use variables (i.e., labels on the unhappy state) to distinguish the different kinds of disputes as well as the corresponding sanctions (Fig. 4).

Note that once the expected behaviors of the contracting parties are modeled by timed automata, we can verify some security properties and enforce them at run-time by checking [6] the execution of the system at runtime according to the scenarios specified by the model. In particular, we can (1) verify statically if the system can reach a dispute state, (2) maintain an audit log and perform model-checking during runtime, and (3) notify the concerned parties in case of contract violation.

This is important in our context where the collaborating organizations are in mutual suspicion. In fact, as there is no "total confidence" between them, the security policy of the service-providing organization discards any request that does not correspond to a signed contract. This is a first level of security. The second level is enforced by runtime model checking. Hence, even if a user succeeds in bypassing his organizations security policy and interacts in a wrong way (accidental or malicious) with another organization (e.g., by

using an authorized Web service in a way which is in contradiction with the signed contract rules), the forbidden interactions are detected and audited (i.e., logged) at runtime. In the same way, if the providing organization does not satisfy its obligations, our e-contract model checking will detect and audit the corresponding misbehavior.

All these issues will be detailed on an example in the next section, where we present a case study that illustrates how to progressively apply our Poly-OrBAC framework.

4. Application to an electrical power grid

4.1. The CRUTIAL architecture

In order to easily understand how to apply PolyOrBAC in the context of CIIs, we first describe the general architecture (Fig. 5) proposed by the european project CRUTIAL [36] to ensure the protection of interconnected critical information infrastructures, in particular those used in the context of electrical power grids, against malicious as well as accidental threats.

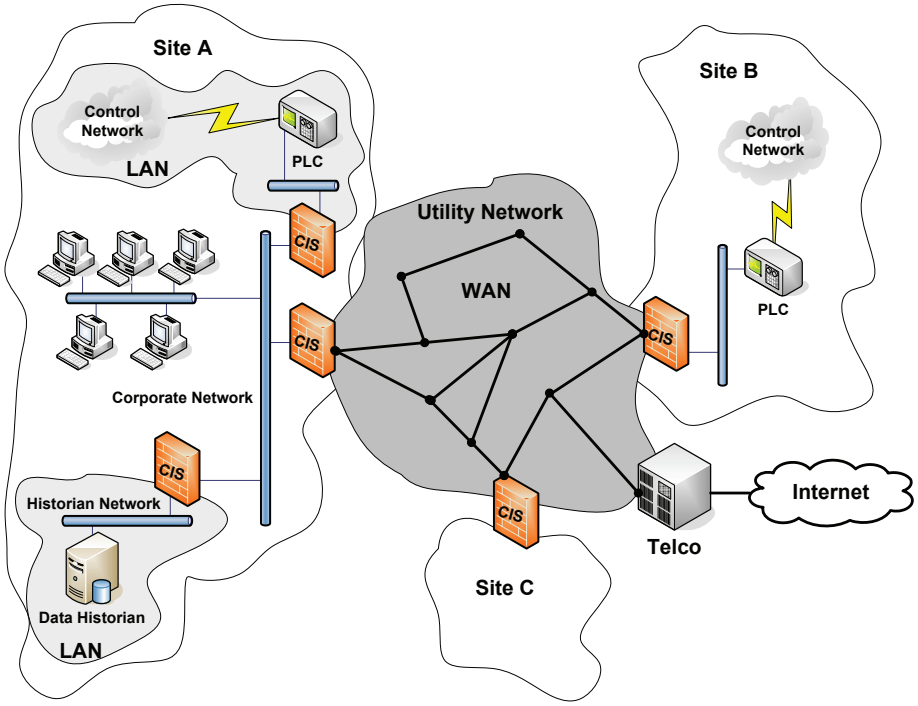

Figure 5: General Architecture of a CII.

The CII architecture can be seen as a WAN of LANs inter-connected by dedicated switches, called CIS (Crutial Information Switch). Each LAN, is composed of logical/physical systems, has its own applications and access control policy, and proposes its services to other systems. Each LAN belongs to a facility (e.g., power plant, substation, control center, etc.), and the WAN interconnects all the facilities belonging to the Critical Infrastructure. The CII is managed and accessed by different actors and stakeholders (power generation, transmission and distribution companies, regulation authorities, communication and computing system providers, brokers, subcontractors, etc.). More than one LAN can be connected by the same CIS if they are part of the same organization and located in the same area. In this case, each LAN is dedicated to a component (e.g., substation), in order to manage a different access control policy for each component.

Considering the CRUTIAL architecture described above as an example, since all organizations of the CII are interconnected by CIS, and in order to provide a controlled cooperation adapted to the CII, each CIS must contain mechanisms to enforce the local security policy of its organization with respect to external accesses. These policies and mechanisms must allow the authorized accesses to the resources and prevent the unauthorized accesses (accidental or malicious ones). In the next subsection we describe one of the scenarios we considered in the CRUTIAL project. Then, we specify its WS and we identify the virtual users. Finally, we present the e-contracts enforcement and runtime checking.

4.2. Electrical Power Grid Architecture and Scenarios

In an electric power grid, one or more electricity generation companies (each in charge of one or several power plants) is connected to one or more transmission grids. Each transmission grid (managed by transmission system operators "TSO") is composed of transmission substations (monitored by one national "NCC" per country and several regional control centers "RCC"), and is connected to one or more distribution grids. Finally each distribution grid (managed by distribution system operators "DSO") is composed of distribution substations (monitored by area control centers "ACC"), and distributes electricity to subscribers (industrial, commercial and residential users) over distribution lines [37].

To illustrate the application of PolyOrBAC on the electrical Power grid CII, we study a practical scenario. This scenario considers the possible cascading effects due to ICT threats to the communication channel among

TSO/DSO Control Centers and their substations in emergency conditions (e.g., line overloads). It is assumed that in emergency conditions, the TSO is authorized by the DSOs to activate defense plan actions for performing load shedding activities on the Distribution Grid.

By studying this scenario, we distinguish four important classes of organizations: Transmission System Control Centres (TS CC) that are managed by TSOs, Transmission System Substations (TS SS), Sistribution System Control Centres (DS CC) that are managed by DSOs, and Distribution System Substations (DS SS). Figure 6 details the most important commands and signals exchanged between these organizations in normal and emergency situations.

In normal operation, all DS SSs send various signals and measurements (power, voltage, frequency) to the TS CC via their DS CC (1) and (3). On the other hand, all TS SSs send various signals and measurements to their TS CC (2). The TS CC monitors the Electric Power System and, if a critical situation is detected, elaborates some potentially emergency conditions that could be remedied with opportune load shedding commands applied to particular areas of the distribution grid. In order to actuate the defense action, the TSO asks a DSO to prepare for a possible load shedding up to a certain power (4). The DSO selects which distribution substations (DS SSs) can shed the needed load with the less severe disturbance for the users and arms these substations for a possible upcoming load shedding (5). When a DS SS receives an arming order from a DSO, it arms the corresponding electrical component (Monitoring Control and Defence Terminal Unit or MCDTU) and sends an acknowledgement to the DSO (6). Once the DS CC has received all the acknowledgements from the DS SS, it sends an acknowledgement to the TS CC. In parallel, the TS CC sends an order to the TS SS controlling the distribution area to prepare itself for a possible load shedding in a near future (7). In case of detection of a real emergency situation, the TS SS sends a load shedding command to all DS SSs participating to the emergency plan, and only the previously armed DS SSs will perform load shedding over their MCDTUs (8). During all the duration of the critical situation, to reduce user disturbance, the DSO can choose to disarm some DS SSs and arm others. When the critical situation is solved (i.e., no more load shedding is expected at short term), the DSO reintegrates the disconnected DS SS.

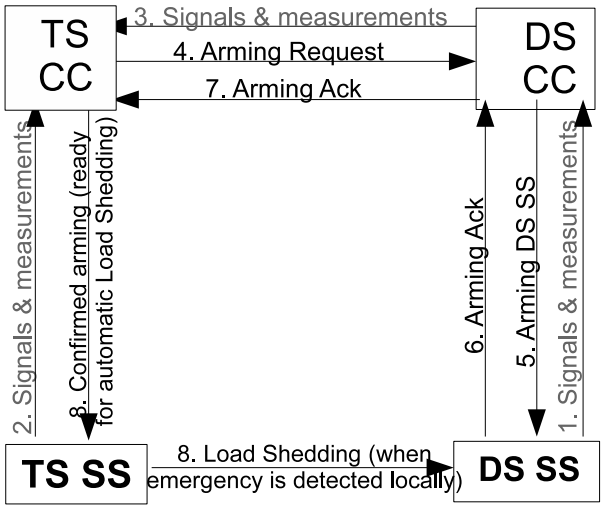

Figure 6: Exchanged commands and signals.

4.3. Electrical Scenario interpretation with PolyOrBAC

4.3.1. Specification of WS and virtual users

In our scenario, we distinguish 4 organizations (TS CC, DS CC, TS SS, DS SS), and three web services (Table 1).

Table 1: Instanciated Web Services

Service	**Provider**	**Client**
WS1-arming-request	DS CC	TS CC
WS2-arming-order	DS SS	DS CC
WS3-prepare-for-LS	TS SS	TS CC
WS4-Load-Shedding	DS SS	TS SS

Figure 7 summarizes the different web services, virtual users (representing remote organizations that can request web services), ws-images (local images of remote web services that can be invoked), and resources involved in the execution of this scenario.

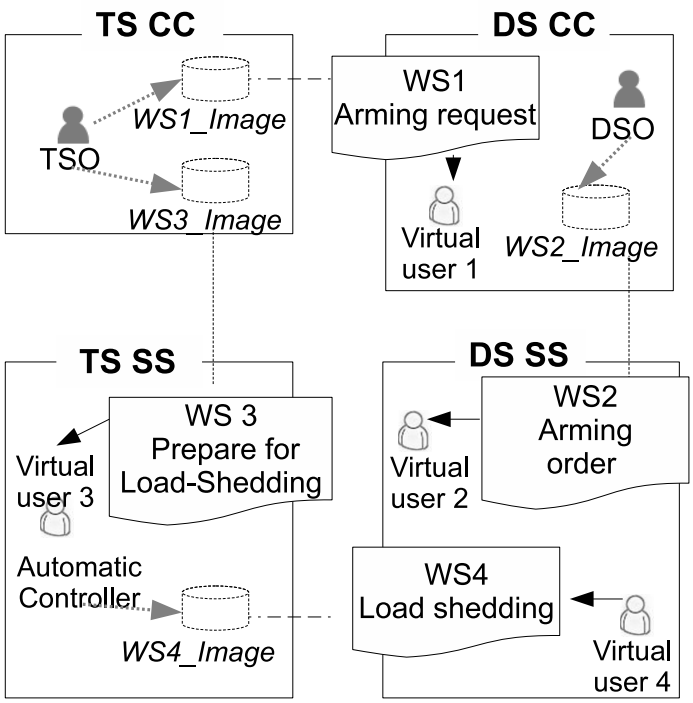

Figure 7: PolyOrBAC approach applied to the electrical scenario.

We assume that the TSO requests the DS CC to arm its DS SS MCDTUs. When the TSO activates WS1-Image, the execution of WS1-arming-request is automatically activated. This access (TSO to WS1-image) is checked according to TS CC policy and is granted according to the OrBAC rule described in table 2, that manages the Access Control for the Arming Request Web service at the level of the organization that invokes the service (i.e., TS CC).

Table 2: Arming Request OrBAC rules at TS CC

Rules
Permission(TS CC, TSO, DS CC arming request, send, critical situation) ∧ Empower(TS CC, Martin, TSO) ∧ Consider(TS CC, invoke_WS1, send) ∧ Use(TS CC, WS1-Image, DS CC arming request) ∧ Hold(TS CC, Martin, invoke_WS1, WS1-image, critical situation) ∧ ⇒ is-permitted(Martin, invoke_WS1, WS1-image)

On the DS CC side, WS1-arming-request asks DSO to arm some DS SS, i.e., to access object WS2-Image. These accesses (virtual-user1 to DSO display, DSO to WS2-Image) are checked according to DS CC policy and is granted according to the OrBAC rule described in table 3, managing the Access Control for both Arming Request and Arming Order Web services at the level of the organization that provides the arming request service (i.e., DS CC).

Table 3: Arming Request/Order OrBAC rules at DS CC

Rules
Permission(DS CC, DSO, DS CC arming order, Send, critical situation) ∧ Empower(DS CC, virtual-user1, DSO) ∧ Consider(DS CC, invoke_WS2, Send) ∧ Use(DS CC, WS2-Image, DS CC arming order) ∧ Hold(DS CC, virtual-user1, invoke_WS2, WS2-Image, critical situation) ∧ ⇒ is-permitted(virtual-user1, invoke_WS2, WS2-Image)

When DSO accesses object WS-Image2, WS2-arming-order is automatically activated, then virtual-user2 activates Object-arm-MCDTU in DS SS, and finally the physical arming command is executed over the physical component MCDTU. This access (virtual-user2 to Object-arm-MCDTU) is checked according to DS SS policy and is granted according to the OrBAC rule decribed in table 4, managing Access Control for Arming Order web service in DS SS. WS3-prepare-for-Load-Shedding, WS4-Load-shedding, WS2-Disarming and WS1-Restart are negotiated and activated in the same way. For simplicity, the two last WSs are not included in table 1.

Table 4: Arming Order OrBAC rules at DS SS

Rules
Permission(DS SS, DSO for SS, Access, DS SS Distrib. Circuits, emergency) ∧ Empower(DS SS, virtual-user2(Subject), DSO for SS) ∧ Consider(DS SS, activate(action), Access) ∧ Use(DS SS, object-arm-MCDTU(object), DS SS Distrib. Circuits) ∧ Hold(DS SS, virtual-user2, activate, object-arm-MCDTU, emergency) ∧ ⇒ ispermitted(virtual-user2, activate, object-arm-MCDTU)

4.3.2. E-contracts enforcement and runtime checking

For each negotiated WS, we specify two automata representing the established contract on the WS client and WS provider sides respectively. In this subsection, we explain how the WS1-arming-request automaton is specified for each side, and how the WS1 client and provider automata are checked at runtime by the corresponding CIS. The automata of the other WS are given in the Annex.

According to the WS1 contract, and as illustrated in Fig. 8, at the TS CC side (the WS1 client) the WS1 automaton waits for an arming request invocation (coming from the TSO). When this invocation is intercepted, the corresponding transition (WS1-arming-request) is activated in the automaton, a timer t is initialized, and the WS1 automaton of TS CC arrives to a state where it waits for a "WS1-arming-request-ack". This acknowledgement should be sent by the DS CC. If the timeout expires without receiving the acknowledgment from the DS CC, a "WS1-arming-request-error" message triggers the exception corresponding to this situation. This exception will be handled by the specific automaton "TS CC-error-handling. Conversely, in normal situations, when the TS CC receives the "WS1-arming-request-ack", its automaton moves to the state where it is ready for an emergency action.

In this state, when the critical situation disappears, the TSO can decide to disarm the distribution substations that were armed. Then, the TS CC sends the "WS1-disarming-request" to the DS CC, and waits for the "WS1-disarming-request-ack" from the DS CC. If the timeout (set to 10 time units in Fig. 8) expires without receiving the "WS1-disarming-request-ack", a "WS1-disarming-request-error" will be sent and then will be handled by the specific automaton "TS CC-error-handling".

Let us now analyse how the DS CC WS1 automaton (the WS1 provider side) reacts to invocations coming from the client side (Fig. 9). Actually, this automaton is waiting for the "WS1-arming-request" from the TS CC; when it receives it, a timer is initialized. Then the DS CC is in a state where it can arm some specific DS SSs. Note that these substations are chosen by the DSO (using the WS2-arming-order).

If the arming operation is not processed after the expected timeout, a "WS1-arming-request-error" is sent, and then, is handled by the "DS CC-error-handling" automaton. If the DS CC is ready for emergency actions, it sends "WS1-arming-request-ack" and moves to the DS CC armed state. If the WS1 automaton of the DS CC side receives a "WS1-disarming-request"

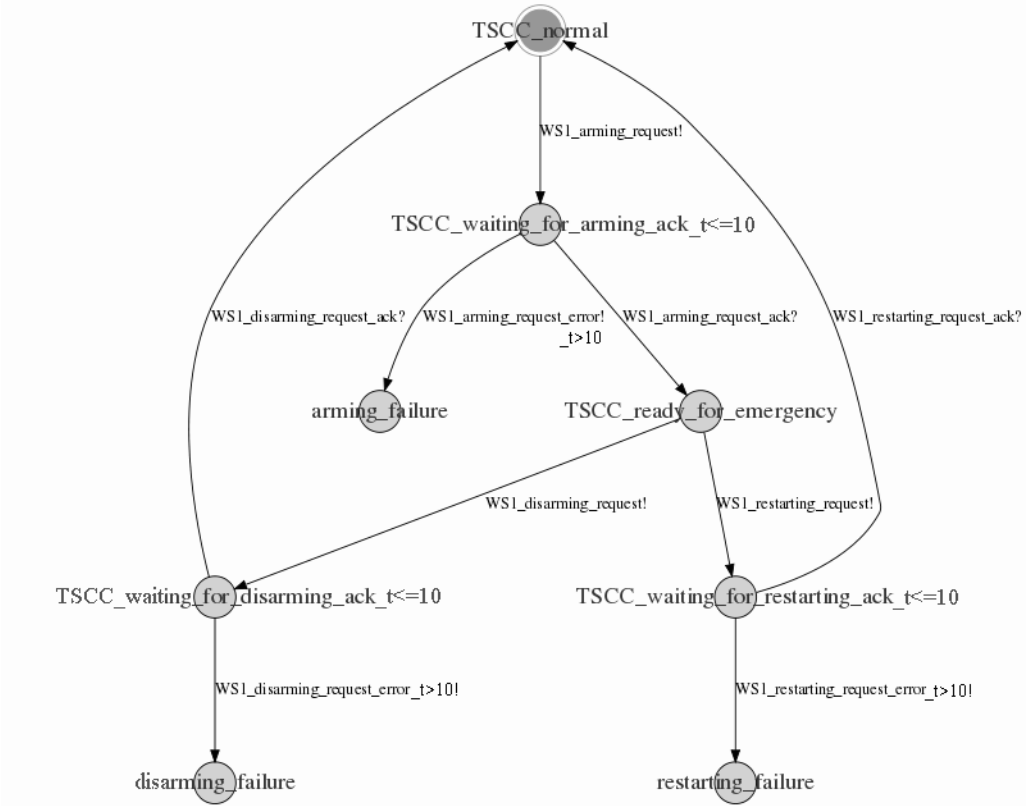

Figure 8: WS1-arming-request automaton in TS CC.

from the TS CC, the disarming operation is carried out and an acknowledgment (the "WS1-disarming-request-ack") is sent to the TS CC. In abnormal situations, if the arming is not processed after the timeout expiration, a "WS1-disarming-request-error" is sent and handled by the "DS CC-error-handling" automaton. Note that disarming activity occurs when there is (2) no emergency situation and (2) no load shedding activity is carried out. Conversely, restarting activity occurs when (1) a load shedding is already carried out and (2) emergency situations disapeared.

5. Implementation

In order to illustrate the feasibility of the scenarios investigated in the previous sections with PolyOrBAC, we have developed a prototype implement-

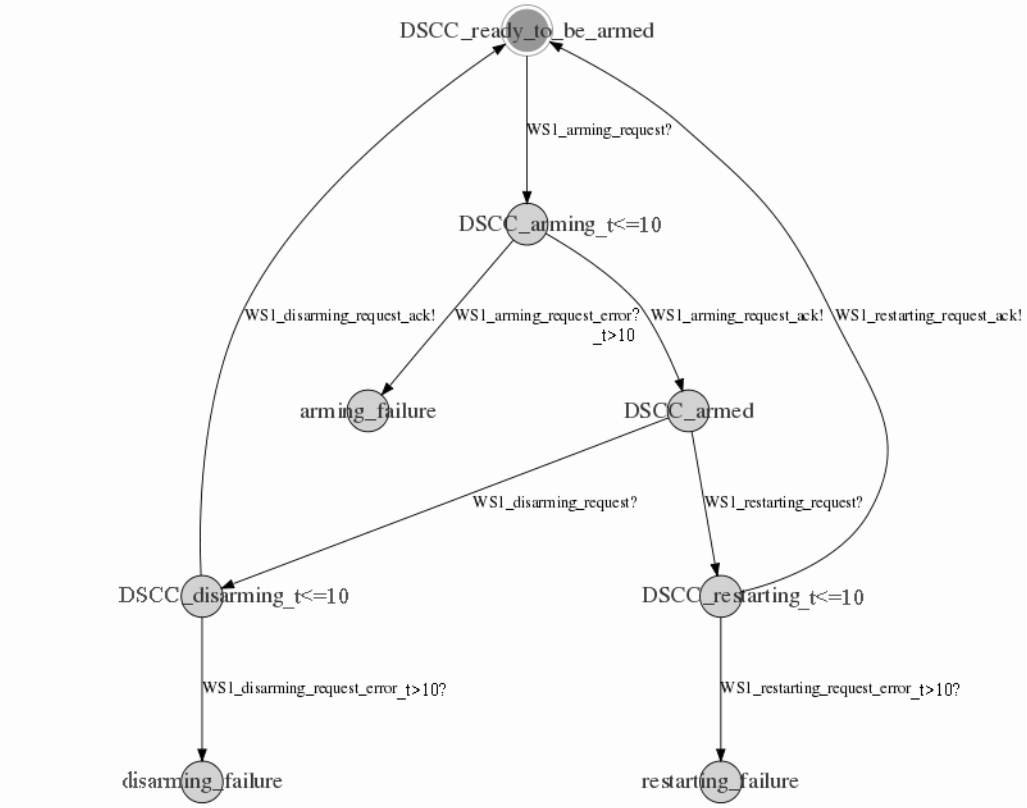

Figure 9: WS1-arming-request automaton in DS CC.

ing and simulating the web services as well as the e-contracts enforcement. We use a decentralized architecture where each organization is responsible for its users authentication and locally controls the access to its own resources and services, according to the WS security architecture (i.e., with PDPs, PEPs, PAP and PIPs [38]). Basically, after receiving an access request (i.e., a web service invocation) from a contracting organization, the provider's Policy Enforcement Point (PEP), located in its CIS, sends the request to the Policy Decision Point (PDP), which evaluates the request and sends back a response. The response can be either access permitted or denied, with the appropriate obligations. The PDP comes to a decision after evaluating the relevant policies. To get the policies, the PDP uses the PAP (Policy Access Point) to extract the security rules (e.g., *Permission*(*Organization*, *Role*, *View*, *Activity*, *Context*)). The PDP may also invoke the Policy In-

formation Point (PIP) service to retrieve the attribute values related to the organization, the subject, the web service (resource), or the environment (the context). This consists in evaluating the associations *Empower* (*org*, *s*, *r*), *Consider* (*org*, *a*, *activ*), *Use* (*org*, *o*, *v*) and *Hold* (*org*, *s*, *a*, *o*, *c*). The authorization decision is sent by the PDP to the PEP. The PEP fulfils the obligations and, based on the authorization decision sent by the PDP, either permits or denies access.

Actually, for simplicity reasons, we made the choice to simulate the whole network using OpenVZ, an open software providing virtualization with high performances (compared to other solutions such as VMware and Xen).

To implement and deploy our architecture, we used the J2EE specification, in particular its Java RMI (*Remote Method Invocation*), Servlet and Axis technologies. Indeed, to manage and deploy web services, we installed an Apache Tomcat server and an Apache Axis Web Service Framework on the machines proposing web services (e.g., DS CC, TS SS and DS SS). Apache Tomcat provides a classical web server as well as an implementation of Servlet and JSP containers. Apache axis is an XML based Web Service Framework, i.e., it allows: (1) deploying web services (using a "Web Service Deployment Descriptor" (WSDD) xml file), (2) generating "Web Service Deployment Language" (WSDL) xml files. Moreover, as web service invocations must be intercepted by the PEP, we decided to deploy a web service on the CIS to dispatch every incoming and exiting calls; the RMI technology handles remote communications.

Besides that, OrBAC rules and relationships are expressed in XML files while Web services are implemented by classes.

Furthermore, JSP is used to implement Web Interfaces and Servlets are used to manage WS calls as well as interactions with CISs.

To verify that interactions are carried out according to the signed contracts, we implemented a dedicated runtime model checker with `UPPAAL`, [39] [40]. UPPAAL automatically converts our automata to XML files that we used in our Java program. UPPAAL also allows proving that the exchange protocol can be run according to the contract clauses.

Note that UPPAAL can also be used to verify properties expressed in a subset of timed Computational Tree Logic (`CTL`) [41]. In our case, we can verify if all the possible executions of the system will never lead to a conflicting situation, which is actually equivalent to verifying the *reachability* property (note that we can also verify the *safety* and *liveness* properties of UPPAAL formulae). These properties are often used in designing a

model to perform sanity checks. A reachability property will validate the basic behavior of the model. For example, the following property "`E <> organization.Dispute`" (expressed in CTL) stands for: "*it exists at least one execution where the organization reaches the dispute state*". Inversely, the property: "`A[] not organization.Dispute`" means that "*none of the possible executions will lead the organization to a dispute state*".

6. Conclusions

This paper presents PolyOrBAC, a security framework which meets the requirements of access control and collaboration of CIIs. Actually, several works have investigated security in workflow and collaborative systems. However, up to our knowledge, none of these works have defined an homogeneous peer-to-peer approach going from the specification to the deployment and runtime checking. Moreover, none of them have applied it to secure CII. Dealing with these issues, our PolyOrBAC framework manages collaboration and resources sharing between all organizations of a CII thanks to the web services technology, while controlling that the interactions between these organizations are in conformity with their needs and their internal security policies specified thanks to OrBAC. Moreover, PolyOrBAC supports the enforcement, the real-time checking as well as the auditing of the exchanges that are established between the different organizations participating in a CII. To check the feasibility of our approach, we also applied PolyOrBAC to a real electric power grid scenario and we presented details of our implementation. The functional aspects are implemented by the Java language, while verification aspects are delegated to a UPPAAL engine.

Our approach can be extended by taking into account availability and integrity requirements. Availability can be handled by means of obligation rules, making mandatory to provide enough resources to achieve the requested activities, even in case of events such as component failures or attacks. For integrity, our approach can be extended to monitor information flows, and prevent flows from lower criticality tasks to higher criticality tasks, except when such flows are validated by means of adequate fault-tolerance mechanisms [42].

Besides that, some recent works have shown how delegation and separation of duties can be expressed in OrBAC, especially using the "context" entity [43]. Other works introduced the recommendation modality and have shown its importance in the CII field [44]. Our work can naturally be ex-

tended to use this expressiveness in order to take into account this kind of notions.

Finally,note that our work can not only be applied to CIIs but also to other collaborative systems, especially thoses with mutual suspious constraints.

Acknowledgments

This work is partially supported by the European FP6-IST research project CRUTIAL (CRitical UTility InfrastructurAL Resilience), the European Networks of Excellence ReSIST and NewCom+, the LAAS project PolSec and the Airbus ADCN+ project. We would like to thank Giovanna Dondossola and Fabrizio Garrone for their contribution to the definition of the case study considered in this paper.

The authors thank the anonymous reviewers for their constructive remarks that helped in improving this paper.

References

[1] S.M. Rinaldi, J.P. Peerenboom, T. K. Kelly, "Identifying, understanding, and analyzing critical infrastructure interdependencies", *IEEE Control Systems, Control of Complex Networks, vol. 21, no 6*, pp. 11–64, 2001.

[2] J. C. Laprie, K. Kanoun and M. Kaâniche, "Modelling Interdependencies Between the Electricity and Information Infrastructures", *SAFECOMP*, Nuremburg, Germany, Springer, LNCS 4680/2008, pp. 57-67, 2007.

[3] A. Massoud, "North America's Electricity Infrastructure: Are We Ready for More Perfect Storms?", *IEEE Security and Privacy 1(5)*, pp. 19–25, 2003.

[4] A. Abou El Kalam, Y. Deswarte, A. Baina and M. Kaâniche, "Access Control for Collaborative Systems: A Web Services Based Approach", *IEEE Intl Conf. on Web Services (ICWS)*, Salt Lake City (Utah, USA), pp. 11–64, 9-13 July 2007.

[5] A. Baina, A. Abou El Kalam, Y. Deswarte, "A Collaborative Access Control Framework for Critical Infrastructures", Proc. of the *Critical Infrastructure Protection* II, Springer, M. Papa and S. Shenoi Editors, pp. 189-201, 2008.

[6] A. Abou El Kalam, Y. Deswarte, "Critical Infrastructures Security Modeling, Enforcement and Runtime Checking", *3rd International Workshop on Critical Information Infrastructures Security* (CRITIS), October 13-15, 2008, Frascati, Italy, to appear in Lecture Notes in Computer Science, Springer, 2009.

[7] A. Abou El Kalam, Y. Deswarte, "Poly-OrBAC: An Access Control Model fior Inter-Organizational Web Services", *Handbook of Research on Semantic Technologies and Web Services*, IGI-Global, ISBN: 978-1-60566-650-1, 2009.

[8] V. Alturi E. Bertino, E. Ferrari, "The specification and enforcement of authorization constraints in workflow management systems". *ACM Transactions on Information and System Security*, 1999.

[9] V. Atluri N.R. Adam and W-K. Huang, "Modeling and analysis of workflows using petri nets", Journal of Intelligent Information Systems, Special Issue on Workflow and Process Management, 1998.

[10] D. Lin, P. Rao, E. Bertino, N. Li, J. Lobo, "Policy decomposition for collaborative access control", 13th *ACM symposium on Access control models and technologies*. Estes Park, CO, USA, ACM Digital Library, 2008, pp. 103-112.

[11] E. Bertino, S. Jajodia, P. Samarati, "Flexible support for multiple access control policies", *ACM Transaction on Database Systems*, 26(2), 214-260.

[12] M. Lorch,S. Proctor, R. Lepro, M. Field, S. Shah, "First experiences using XACML for access control in distributed systems", *ACM workshop on XML security*, Fairfax, Virginia, 2003.

[13] C. Sturm, K.R. Dittrich, P. Ziegler, "An access control mechanism for P2P collaborations". *international workshop on Data management in peer to peer systems*, Nantes, France: ACM, pp. 51-58.

[14] M. Shehab, E. Bertino, A. Ghafoor, "Secure collaboration in mediator-free environments", 12th *ACM conference on Computer and communications security*. Alexandria, VA, USA, ACM digital Library, pp. 58-67, 2005.

[15] L. Pearlman, V. Welch, I. Foster, C. Kesselman, "A Community Authorization Service for Group Collaboration", *Third International Workshop on Policies for Distributed Systems and Networks*, IEEE computer society, pp. 50-59, 2002.

[16] A. Abou El Kalam, Y. Deswarte, "Multi-OrBAC: a new access control model for distributed, heterogeneous and collaborative systems", *IEEE Symp. on Systems and Information Security*, Sao Paulo, Brazil, 2006.

[17] F. Cuppens, N. Cuppens-Boulahia, C. Coma, "O2O: Virtual Private Organizations to Manage Security Policy Interoperability", *Second International Conference on Information Systems Security* (*ICISS* 2006), India, December 17-21, 2006.

[18] A. Abou El Kalam, R. El Baida, P. Balbiani and S. Benferhat, F. Cuppens, Y. Deswarte, A. Miege, C. Saurel and G. Trouessin, "Organization Based Access Control", *Proc. of IEEE 4th Intl Workshop on Policies for Distributed Systems and Networks (POLICY)*, Lake Come, Italy, (Utah, USA), pp. 120–134, June 2003.

[19] R. S. Sandhu, E. J. Coyne, H. L. Feinstein and C. E. Youman, "Role-Based Access Control Models", *IEEE Computer 29(2)*, pp. 38–47, February 1996.

[20] D. Ferraiolo, R. S. Sandhu, S. Gavrila, D. Kuhn, and R. Chandramouli, "A Proposed Standard for Role-Based Access Control", *ACM Transactions on Information and System Security, v 4, n 3*, 2001.

[21] W3C, XML, *W3C Recommendation*, February 2004.

[22] W3C, SOAP, *W3C Recommendation*, June 2003.

[23] W3C, WSDL, *W3C Candidate Recommendation*, March 2006.

[24] OASIS, UDDI, *UDDI Specifications TC*, February 2005.

[25] K. Beznosov, Y. Deng, "A Framework for Implementing Role-Based Access Control Using CORBA Security Service", *4th ACM Workshop on Role-Based Access Control*, October 28-29, 1999, Fairfax, VA, USA, pp 19-30.

[26] N. Vuong, G. Smith, Y. Deng, "Managing Security Policies in a Distributed Environment Using eXtensible Markup Language", ACM Symposium on Applied Computing, Las Vegas, Nevada, United States, pp 405-411, 2001.

[27] X. Feng, L. Guoyuan, X. Xuzhou, "Role-based Access Control System for Web Services"', *4th International Conference on Computer and Information Technology*(CIT'04),Wuhan, China,14-16 September 2004, pp. 357-362 .

[28] K. Leune, W. Van, H. Heuvel, "A Methodology for Developing Role-Based Access Control to Web-Services", Infolab Technical Report Series, no. 11, December 2002.

[29] OASIS, "XACML profile for Role Based Access Control"', Committee Draft 01, 13 February 2004.

[30] R. Alur, D. L. Dill, "A theory of Timed Automata", *Theoretical Computer Science*, 126(2): 183-235, 1994.

[31] C. Bettini, S. Jajodia, X. S. Wang et D. Wijesekera, "Obligation Monitoring in Policy Management", *International Workshop on Policies for Distributed Systems and Networks* (Policy), Monterey, California, 5-7 June 2002, IEEE Computer Society Press, pp. 2-12.

[32] N. Demeanor, N. Delay, E. Lupus, M. Sloan. "The Ponder Policy Specification Language", *International Workshop on Policies for Distributed Systems and Networks*, Bristol, UK, IEE Computer Society Press, pp.18-38, 2001.

[33] Q. Ni, E. Bertino, J. Lobo, "An Obligation model bridging access control policies and privacy policies", 13th *ACM SACMAT*, Estes Park, CO, USA, June 11-13, 2008.

[34] M. Hilty, A. Pretschner, D. Basin, C. Schaefer and T. Walter, "A Policy Language for Distributed Usage Control", 12th *European Symposium*

On Research In Computer Security (ESORICS), Dresden, Germany, September 24 26, 2007.

[35] OASIS, eXtensible Access Control Markup Language TC v2.0, Normative XACML 2.0 documents.

[36] Verissimo, N.F. Neves, M. Correia, Y. Deswarte, A. Abou El Kalam, A. Bondavalli, A. Daidone, "The CRUTIAL Architecture for Critical Information Infrastructures", in *Architecting Dependable Systems V*, Springer, LNCS 5135, 2008, pp. 1-27.

[37] F. Garrone, C. Brasca, D. Cerotti, D. Codetta Raiteri, A. Daidone, G. Deconinck, S. Donatelli, G. Dondossola, F. Grandoni, M. Kaaniche and T. Rigole, "Analysis of new control applications", *CRUTIAL project, Deliverable D2*, January 2007.

[38] Organization for the Advancement of Structured Information Standards (2006). *OASIS Web Services Security TC, Web Services Security v1.1 (WS-Security)*, February 2006.

[39] UPPAAL, tool available at http://www.uppaal.com

[40] K.G. Larsen, P. Pettersson, W. Yi, "UPPAAL in a nutshell", *Journal of Software Tools for Technology Transfer*, 1(1-2): 134-152, 1997.

[41] B. Berard, M. Bidiot, A. Finkel, F. Larousinie, A. Petit, L. Petrucci, Ph. Schnoebelen, P. McKenzie, *Systems and Software Verification, Model Checking Techniques and Tools*, Springer, 2001, ISBN 3-540-41523-8.

[42] E. Totel, J.P. Blanquart, Y. Deswarte and D. Powell, "Supporting multiple levels of criticality", *28th IEEE Fault Tolerant Computing Symposium (FTCS-28)*, Munich (Germany), pp. 70–79, June 1998.

[43] M. Ben Ghorbel, F. Cuppens, N. Cuppens-Boulahia and A. Bouhoula, "Managing Delegation in Access Control Models". 15th *International Conference on Advanced Computing and Communication* (ADCOM'07), Guwahati, India, December 18-21, 2007.

[44] A. Abou El Kalam, P. Balbiani, "A Policy Language for Modelling Recommendations", *IFIP TC-11 International Information Security Conference*, (IFIP SEC 2009), Cyprus, May 18-20, 2009, Springer.

Annex

A. WS2-arming-order automata

At the DS CC side, to carry out DS SS arming, the "WS2-arming-order" is sent to all the DS SS that need to be armed.

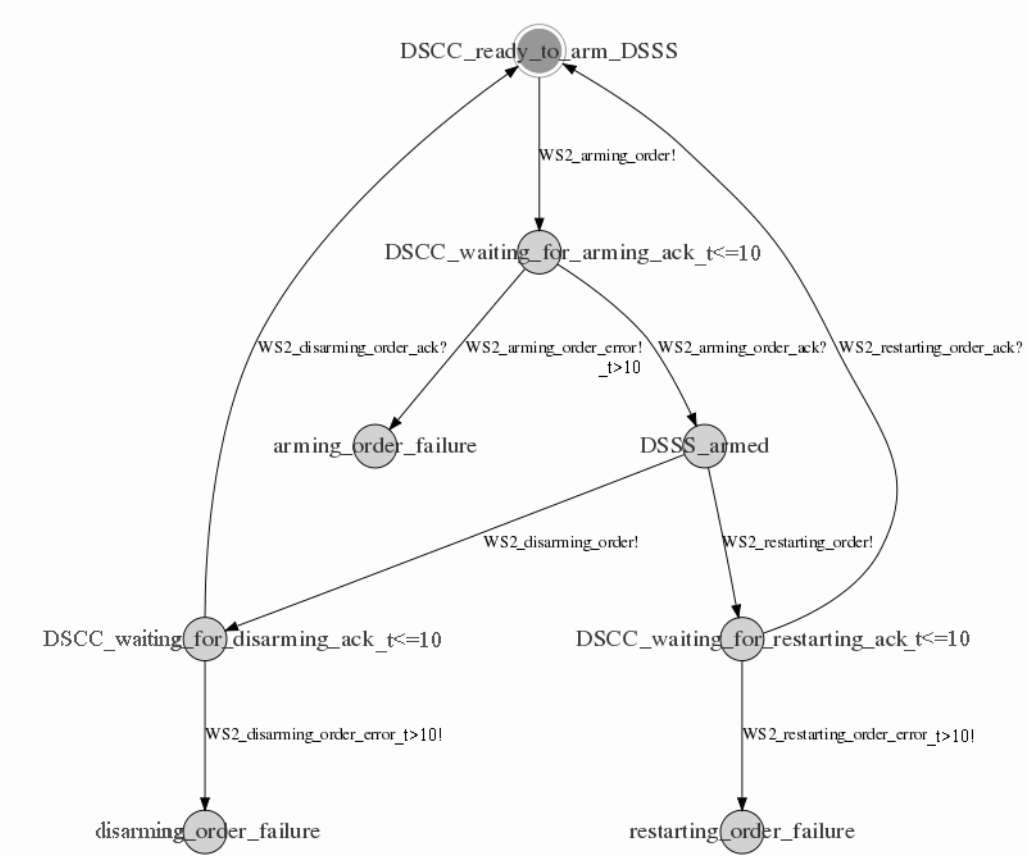

Figure 10: WS2-arming-order automaton in DS CC.

A timer is then initialized and the WS2 automaton of DS CC (is in a state where it) waits for arming acknowledgments ("WS2-arming-order-ack") coming from each armed DS SS. If the timeout expired without receiving the acknowledgments, a "WS2-arming-order-error" is sent and is handled by the "DS_CC-error-handling" automaton. Conversely, in normal situations, when the DS CC receives the "WS2-arming-order-ack", it become ready for

emergency actions, and a "WS1-arming-request-ack" is finally sent back to the TS CC.

Under certain situations, the DSO may decide to disarm some armed substations. In that case, it sends a "WS2-disarming-request" to each DS SS to disarm. Then, the DSCC WS2 automaton waits for the "WS2-disarming-request-ack" from the DS SS. If the timeout expired without receiving the acknowledgment ("WS2-disarming-request-ack"), a "WS2-disarming-request-error" is sent and is handled by the corresponding automaton ("DS_CC-error-handling").

Figure 11: WS2-arming-order automaton in DS SS.

At the DS SS side (Figure 11), the WS2 automaton waits for the "WS2-arming-order". When the DS SS arming is carried out, an acknowledgment

("WS2-arming-order-ack") is sent to the DS CC; the DS SS is now armed. Afterwards, if the DSSS receives a "WS2-disarming-order", it performs the disarming operation and sends back the "WS2-disarming-order-ack" to the DS CC.

B. WS3-prepare-for-Load-Shedding automata

As indicated in Figure 12, in emergency situations, when the TSO decides to launch the load shedding activity, he actually sends a prepare-for-LS invocation to the TS CC; the latter invokes "WS3-prepare-for-Load-Shedding" of the TS SS.

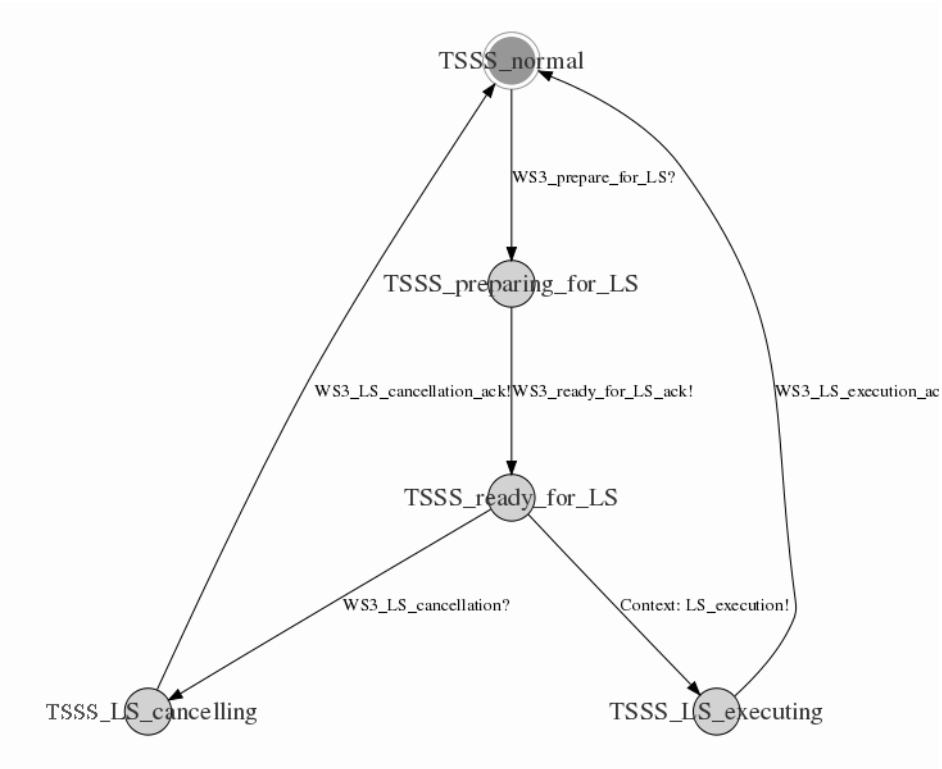

Figure 12: WS3-loadshedding automaton in TS SS.

Afterwards, if the TSO decides to cancel the loadshedding preparation, he sends a LS-cancellation to the TS CC, which then sends the "WS3-

Load-Shedding-cancellation" to the TS SS and waits for the acknowledgment ("WS3-Load-Shedding-cancellation-ack"). At the TS SS side, the WS3 automaton receives the "WS3-prepare-for-LS" synchronization (Fig. 13).

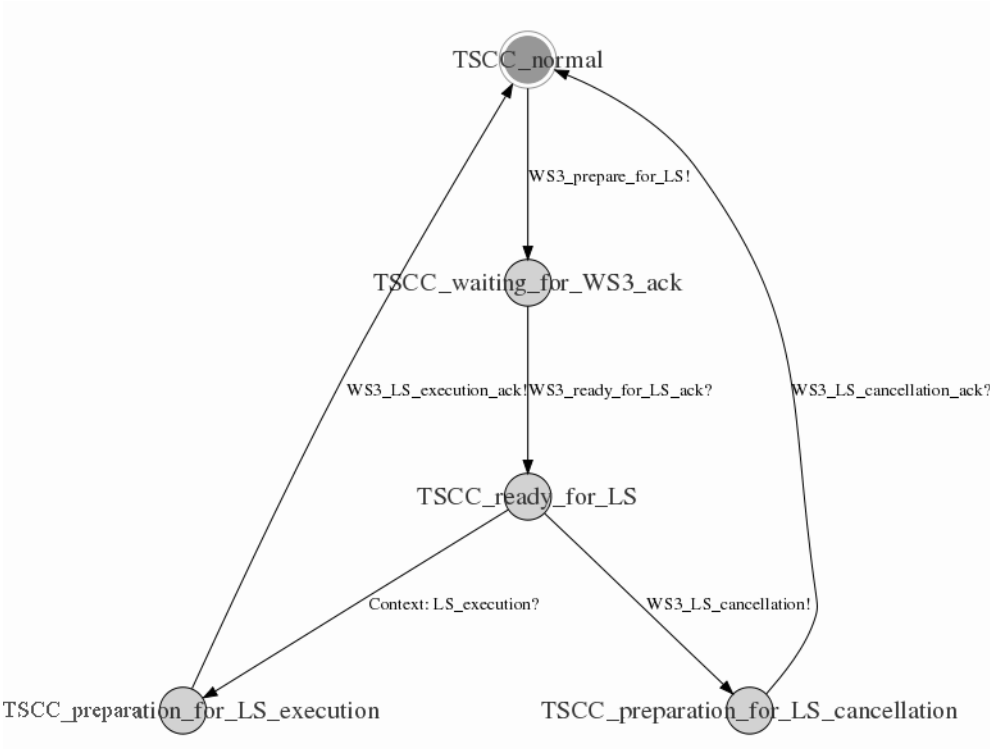

Figure 13: WS3-loadshedding automaton in TS CC.

The load shedding can now be carried out, and the "WS3-ready-for-Load-Shedding-ack" is sent back to the TS CC. If the TSO decides that the pre-emergency situation has disapeared, the TSO can request the TS SS to cancel the load shedding preparation by sending "WS3-prepare-LS-cancellation" and the corresponding acknowledgment ("WS3-Load-Shedding-cancellation-ack") is sent back to TS CC.

C. WS4-load-shedding automata

At the TS SS side, the WS4 automaton (Fig. 14) specifies that when some specific conditions are fulfilled (e.g., when an emergency situation is detected by the TS SS), the TS SS can decide to launch the "WS4-loadshedding" on the armed DS SS.

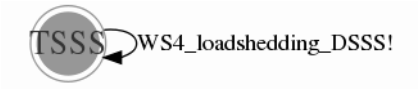

Figure 14: WS4-load-shedding automaton in TS SS.

From the WS4 point of view, TS SS stays in the same unique state (which is not true from the point of view of WS3).

At the DS SS side (Fig. 15) , the loadshedding operation is then automatic, and the DS SS stays in the same state (from WS3 point of view).

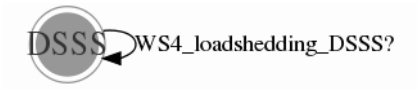

Figure 15: WS4-loadshedding automaton in DS SS.